ISSN 0265-2919

80p 7

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

APPLICATION

SHOWCASE With the right graphics software you can create your own shop window display or broadcast to the nation — 132

HARDWARE

ON THE WRITE TRACK We look at disk formatting and the way information is arranged on floppy disks — 124

DRAGON COMES OF AGE The Dragon 64 updates the original 32 and offers some possibilities for serious business use — 130

SOFTWARE

TALKING BUSINESS Continuing our review of business software and the benefits it can bring the small trader — 121

COMPUTER SCIENCE

THE LOGICAL ANSWER We pause and recap on the course so far — 126

JARGON

FROM BACKGROUND TO BACKUP A week-by-week glossary of computing terms — 129

MACHINE CODE

MEMORY SPACE Machine code programs have to go somewhere in memory. This week we discuss the safe spots — 135

PROFILE

PAGE THE ORIC We look at a young company in a challenging market — 140

WORKSHOP

COMPONENT PARTS We look at the fundamental components and show two ways of calculating resistor values — 138

Next Week

- **MSX is a hardware and software standard designed to remove the greatest problem of the microcomputer industry — that of incompatibility between different systems.**
- **We welcome the return of Programming Projects and take a look at how to handle advanced mathematics in BASIC.**
- **The Spectravideo is the first micro available in this country with many of the features associated with the MSX standard.**

COMING VERY SOON

FREE

MACHINE CODE MONITOR AND ASSEMBLER PROGRAM ON CASSETTE

Written specially for Home Computer Advanced Course readers

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH

Editors Jonathan Hilton, Max Phillips; **Art Director** David Whelan; **Production Editor** Catherine Cardwell; **Staff Writer** Brian Morris; **Picture Editor** Claudia Zeff; **Designers** Hazel Bennington, Julian Dorr; **Sub Editor** Robert Pickering; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Researcher** Helena Siedlecka; **Contributors** Lisa Kelly, Steven Colwill, Geoff Nairn, Geoff Bains, Martin Hayman, Tony Harrington, Richard Pawson; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

TALKING BUSINESS

One of the simplest ways of approaching business systems design is to consider the cash flowing into and out of the business. Suppliers of software who aim their products at small businesses like to take this approach. It enables them to offer a complete applications solution in one, all-embracing program.

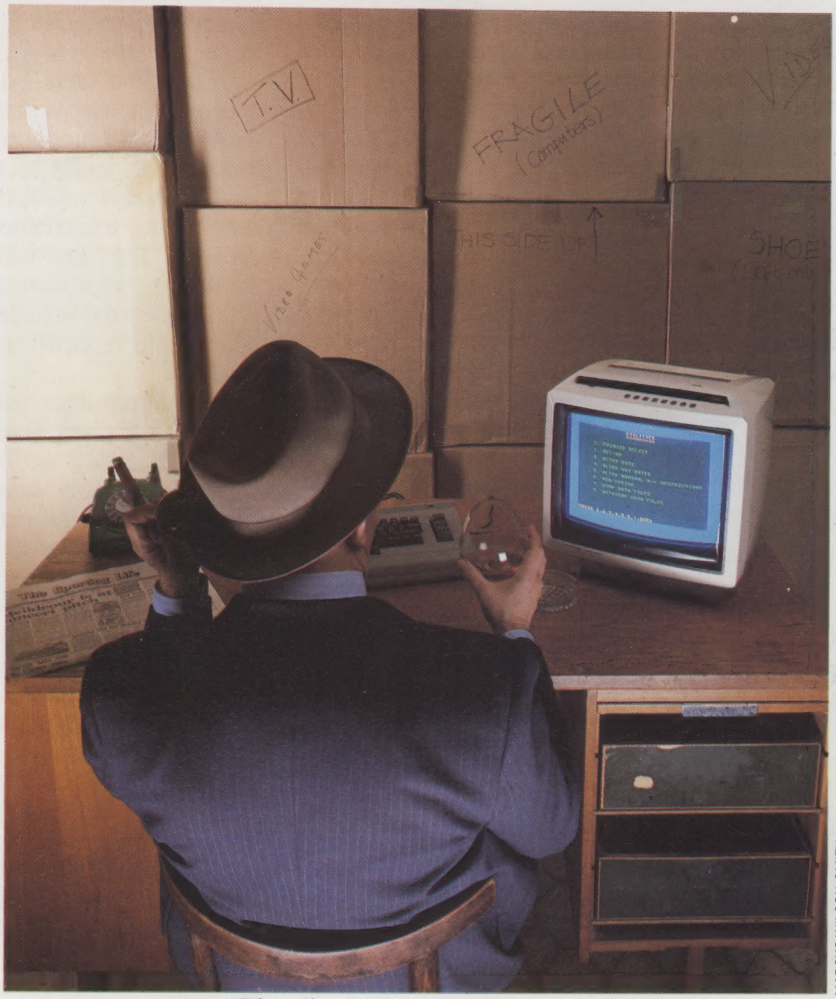
The three programs we have chosen to illustrate the design requirements of a business package are: Quick-Count's Book-keeping System For The Cash Trader, a cassette-based program for the Commodore 64; Accountant, by Compact Accounting Services, running on the BBC Model B Micro equipped with disk drives and Acorn's Z80 second processor; and Microledger, by Lewis Ashley Computers, a disk-based system running on the Apple II.

To provide a complete book-keeping solution, a program has to be able to break down the various sources of income and expenditure in the business. Business people don't just want to know the summary total of how much has been made or spent in any one period. They need to be able to analyse those totals into their component parts. They need to know how much is going on rent, on travel, on petty expenses, on stationery and so forth. Otherwise the only book-keeping system the trader would need would be an adding machine to tot up the incomings and outgoings.

To meet the need for analysis of the cash flow, the program has to give the user a means of allocating income and expenses to various 'account headings'. The *general ledger* is the accounting device that contains these headings. It forms the heart of any business system.

Cash Trader is an excellent example both of what can be achieved with a cassette-based program and of the limitations of such programs. It has a general ledger consisting of 79 different account headings. This is tiny by comparison with disk-based systems, the smallest of which offer hundreds of account headings, but it should be more than enough for most traders to identify all the areas of interest to them in their operation.

Cash Trader's general ledger is *pre-structured* (in contrast to both Accountant and Microledger, which are free format ledgers). 'Pre-structured' means that the range of possible account numbers is already divided up into the different types of general ledger accounts. This makes it easier for the program to decide what operations to carry out on the data in each account for reporting purposes. It also makes it easier for the new user to set up. The disadvantage is that it imposes restrictions on the kind of ledger structure that can be created.



MARCUS WILSON-SMITH

The Cash Trade
Even a cassette-based home micro is suitable for a small business operation. Most of the packages at this level have a narrow focus, such as dealing with cash flow alone, and are not full accounting systems

Numbers 01 to 19, for example, form the trading accounts. In other words they deal with income from sales and purchases (of goods for resale or of stock) made by the business. However, the trader might only want to use four of the possible 19 trading accounts. A single account, say Number 01, might be designated 'Takings' and all the income generated by trading would be entered (the technical term is *posted*) to this account. Account 01 will then keep a cumulative total of all the income.

On the other hand, the trader might decide to keep separate totals for, say, five different categories of goods sold. This would mean opening five income accounts (numbers 01 to 05). It would then be necessary to set up some procedures to ensure that the right sales were posted to the right accounts. But Cash Trader would have no difficulty with this level of detail.

Account numbers 20 to 49 relate to profit and loss accounts. This section is meant to cover all the different kinds of expenses that the business has to meet. The trader will want to create account headings such as: rent, salaries, rates, electricity, bank charges, bank interest, stationery, advertising, and telephone.

Account numbers 50 to 79 relate to balance sheet accounts. These show the overall financial state of the business at any given time. Balance sheet account headings will include, for example, the fixed assets account (showing the value of all the assets owned by the business), the current bank account (or accounts, if there is more than one), the VAT account, and the creditor control account (showing how much is owed to all the business's suppliers).

In a manual book-keeping system, the trader has to write up the sales and purchase daybooks, showing the summary total of all the takings and purchases for the day or the week. The Cash Trader menu for posting values to the nominal ledger is almost self-explanatory:

1. Daily Takings
2. Payments in Cash
3. Payments from Current Bank A/C
4. Journals

Option 1, Daily Takings, is designed to record a week's takings. It asks the user to select a day (numbered 1 to 7) and then to input a summary total for that day's takings. The only differentiation in sales is that the user can flag a total as a 'special item'.

This recognises the fact that not all income comes from ordinary sales. If a garden shop sells its van, that is a special sale and lumping that cash in with the day's takings will produce a distorted sales figure. Cash Trader, rather surprisingly for so simple a program, has the necessary facilities for identifying 'special item' sales. But that is the only analysis or breakdown of the total daily sales that the program allows. The user can choose whether the amount should be posted to one of three nominal accounts: the bank account, cash account or credit card suspense account.

On the payments side (the equivalent of the purchase ledger daybook) the system allows the user to pay from either the cash account or the bank account. A three-digit reference code identifies each payment transaction. And it allows a 16-character narrative description of the reason for each transaction (usually the supplier's name). It also calculates any necessary VAT on payments and posts the amount to the VAT control account.

Anyone using Microledger or Accountant will have to set up a similar sort of structure. A major difference though is that with these two systems there is no pre-defined order to the accounts. With Accountant, for example, the user is given an eight-digit code structure and can allocate whatever numbers are required to any account. Microledger has a three-digit code structure, but the same free format principle applies.

Compact's Accountant program is also based around a nominal ledger — though a much more sophisticated one, capable of dealing with many more account headings. It also has a comprehensive analysis facility. But it forces the user to summarise data entries to a certain extent, in comparison with a full sales and purchase ledger system.

With a full sales ledger and purchase ledger program the user is able to keep a master file of customers and suppliers respectively. This file will include full details of the customer or supplier account and it will also keep a full record of all outstanding transactions with that customer or supplier. The user will be able to call up the account on the screen and be supplied with a full listing of all the invoices posted to that account and all the receipts or payments made to that account.

Accountant does not attempt to provide this. Instead, like Cash Trader, it takes the daybook approach. Being a disk-based system though, with considerably more memory space at its disposal, it does not need to have its information input in such a condensed form.

Instead of inputting just one total sales figure for each day of the week, the user can put through as many entries as necessary. The system recognises five different types of transactions: invoices, credit notes, cash receipts, cash sales and miscellaneous receipts. Each entry can be described with up to 16 characters, given a unique reference number, and analysed to any number of nominal ledger accounts (rather than the simple choice of three offered by Cash Trader).

The purchase daybook can deal with credit or cash purchases and miscellaneous purchases. It also has a detailed VAT reporting facility that keeps track of all the trader's input VAT.

Microledger, in contrast to both these systems, runs a full sales and purchase ledger. The user can create up to 999 separate sales and purchase ledger accounts. Each account will record the customer's name and up to five address lines. The account automatically maintains a cumulative balance and a list of all outstanding transactions.

The major difference between Microledger and the previous two packages described is that it contains a great deal of information about the amount of business done with each customer and supplier. Cash Trader condenses that information into one or two summary totals. Accountant can process individual transactions with customers or suppliers, but the user will need to keep a manual sales or purchase ledger in order to see easily what each customer or supplier balance is.

In the next instalment of this series we will continue our comparison of these three packages, looking at the ways they handle the input of values into nominal account headings. We will also consider the reporting routines that enable the user to see the data in these headings.

Keeping Tabs
Arthur is doing his accounts for the week. He has sold three cases of Glen Kyushu whisky for cash, and 10 cases by credit card; he has received a cheque for £500 for services rendered, and has sold the Rolls Royce for cash. His payments for the week have been cash for more whisky, cash to Terry, his sole employee, and a cheque for a new car

Typical Transaction Entry
All takings are entered this way, and are marked as GOODS or SPECIAL depending on the nature of the sale, and CASH, BANK or CREDIT CARD depending on how the customer paid

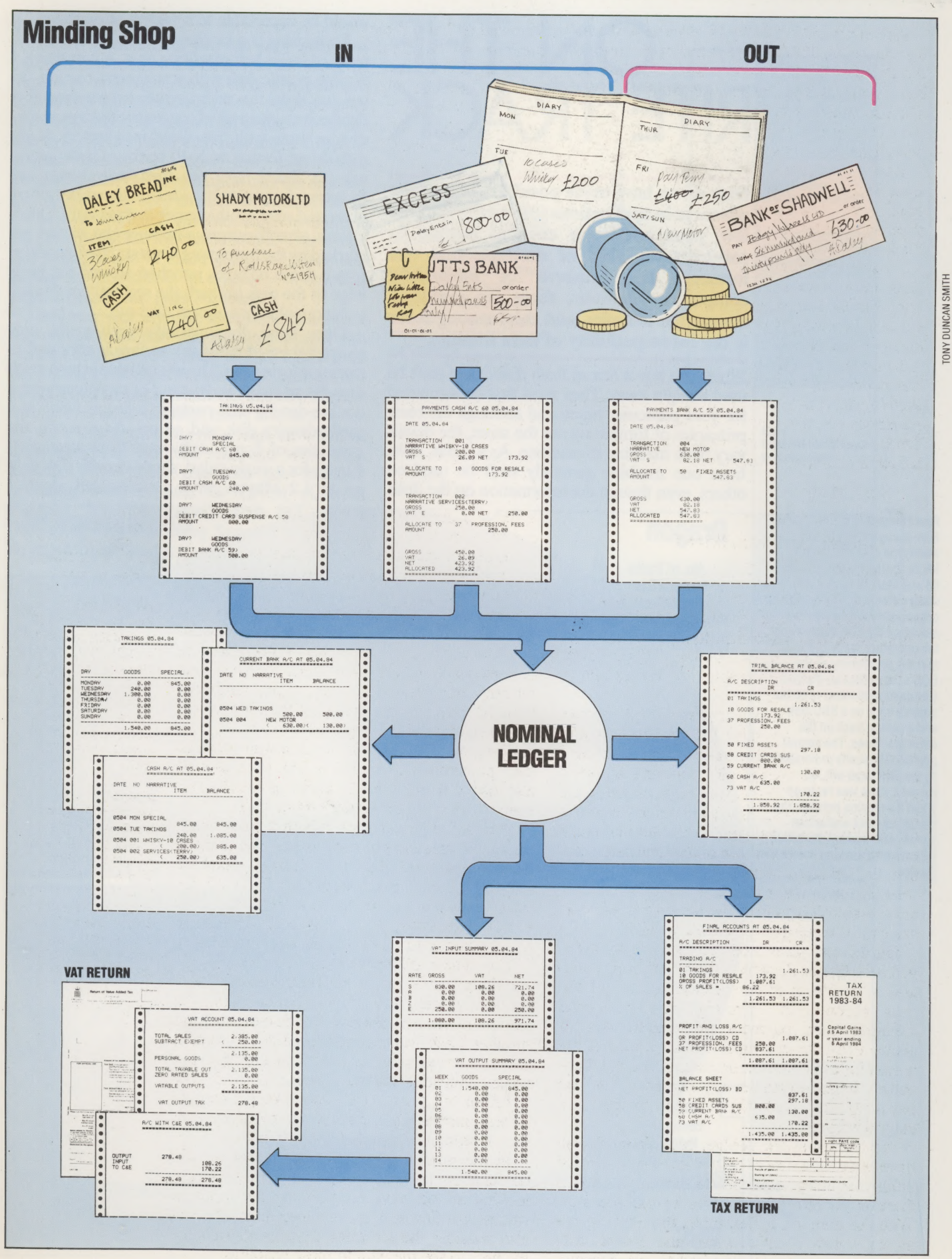
Typical Payments Entry
All payments are entered in the same way, showing CASH or BANK, description, VAT rate, and code number of the ledger account to be debited

Reports
These summaries are generated automatically at the end of a PAYMENTS or TAKINGS entry run

Nominal Ledger
This comprises up to 79 nominal accounts entitled: TAKINGS, STOCK, SALARIES, RENTS, BANK CHARGES, FIXED ASSETS, CASH, etc. These are grouped for balance purposes into the TRADING account, PROFIT and LOSS account, and BALANCE SHEET

Reports
A variety of reports is possible: the state of the CASH account and BANK account and a TRIAL BALANCE are what most traders would want to see after entering a week's, or a day's trading. FINAL account and VAT report would probably be required only every quarter

Minding Shop



TONY DUNCAN SMITH

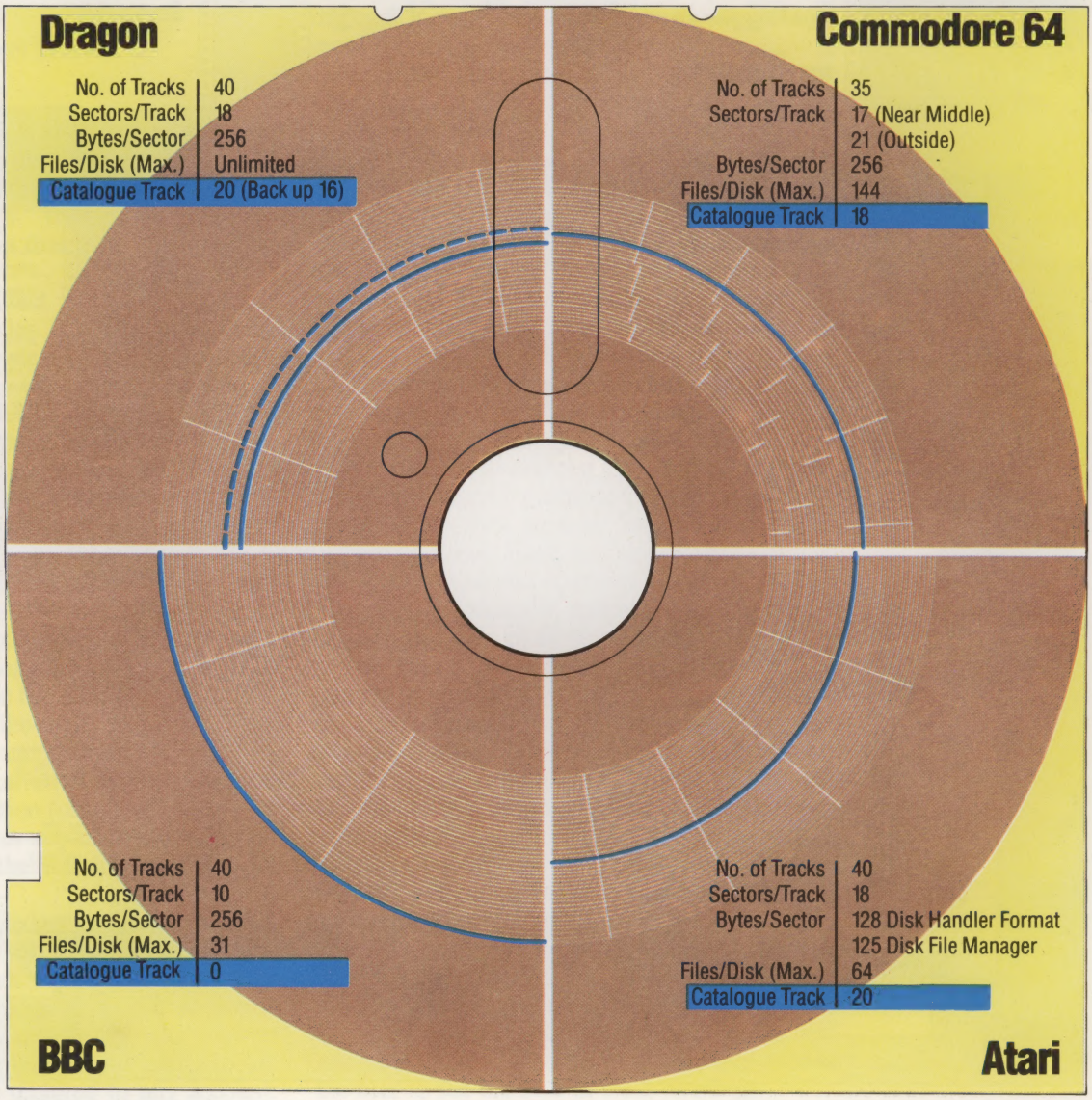
ON THE WRITE TRACK

Disk drives have made possible many of the advances in applications software of the past few years and have enabled anyone to construct the kind of database that was formerly the preserve of mainframe computers. However, there are problems among the benefits, and chief among these is the incompatibility of data formats.

When you buy a box of fresh disks, they can't be used straight away. They must first be formatted for your particular micro and although the basic principles of formatting are the same, the details vary from machine to machine. As a result, disks from one machine generally can't be used on others, even though the information on the disks would be acceptable to a variety of machines.

Formatting is a bit like ruling lines on a blank piece of paper before writing on it. A new disk is a flexible plastic disk with a magnetic coating. To use the disk, the micro writes information that divides it up into a set of concentric tracks and sub-divides these with 'pie-slices' into sectors. Some disk drives format with 40 tracks of data, others wih 80. Some have two read/write heads and therefore format both sides of the disk, others write and read on the upper surface only. This is known as *soft-sectoring* because the sectors are marked by the formatting process. *Hard-sectoring* uses a series of holes punched around the inner edge of the disk to mark the sectors, but this is a technique that has almost died out.

Once these divisions have been made, only about a third of the disk's surface is allocated to storing information. The last variable in how much data fits on a disk is *density*. For any given area, a double-density disk packs in twice as much as a single-density disk and so on. As a result, the capacities of floppy disks vary from around 90 Kbytes for a single-sided, single-density disk right up to 1.2 Mbytes on a double-sided, double-



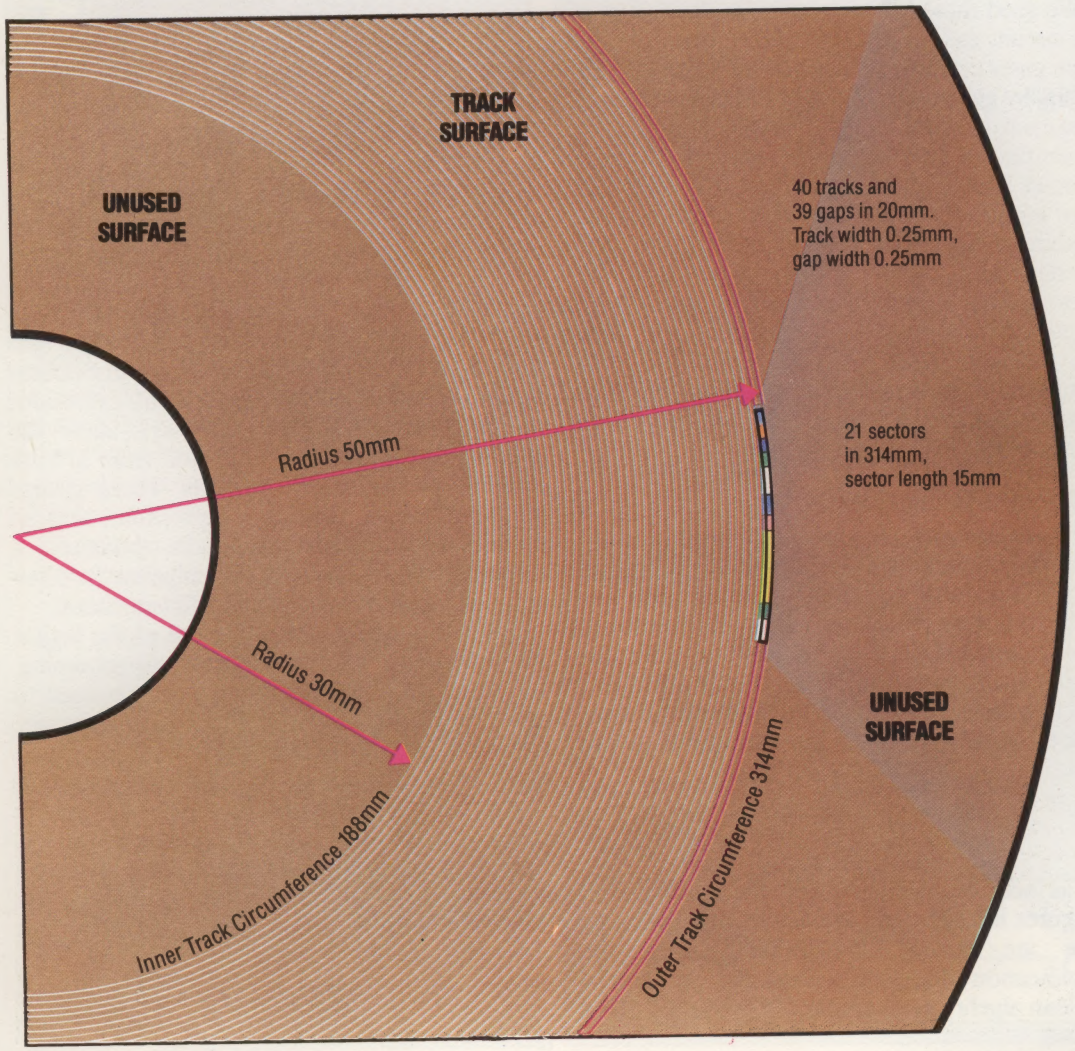
Four Formats
Even though they use the same type of disks, the BBC Micro, Dragon, Commodore 64 and Atari all format their disks differently. The number of separate files that can be stored on the disk depends on the operating system. Notice that the positioning of the DOS's list of files on the disk (the catalogue track) is on the outside edge on a BBC disk and a central track on the other machines. The central position is usually preferable as the disk head will, on average, have less far to go when it's moving between the catalogue track and actual data tracks, making the whole process of accessing the disk quicker

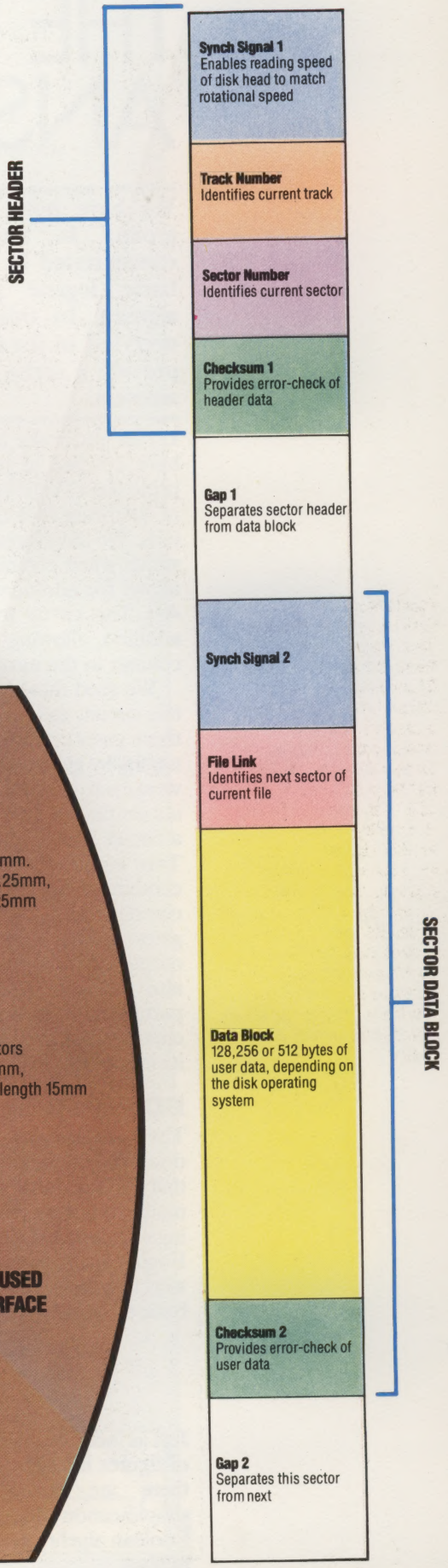
density disk.

Once it is formatted, you can copy programs and information onto the disk. This isn't done on an ad-hoc or straightforward basis. Everything is kept in files and controlled by the disk operating system (DOS). This keeps a list of the files on the disk in a directory and then uses any convenient unused sectors to store the information. Physically, a file can be split up all over the disk and the DOS will also keep an index of which sectors a file uses, leaving markers in each sector as to where the next part of the file can be found. This is a bit like taking a pile of ruled pieces of paper and adding the ingredients of a book — a contents page, an index and page numbers.

Until very recently, disk formatting has been a very complex and non-standard area fraught with incompatibilities. The ability to take a disk from one machine and use it in another depends on two things: how the disk is formatted and how files are arranged on it. Fortunately, many micros now use the same disk formatting or at least have special utility programs that can read and write alien formats. And many machines use standard operating systems such as CP/M (Control Program Microprocessors) and MS – DOS (Microsoft Disk Operating System), so files are arranged the same way.

Tracks And Sectors
Data is written on the disk in concentric tracks. Each track is divided into sectors. A sector contains a block of the user's data, and to this block the DOS automatically attaches the error-checking and identification data necessary for file and record management


TRACK SURFACE
UNUSED SURFACE
40 tracks and 39 gaps in 20mm. Track width 0.25mm, gap width 0.25mm
Radius 50mm
21 sectors in 314mm, sector length 15mm
Radius 30mm
UNUSED SURFACE
Inner Track Circumference 188mm
Outer Track Circumference 314mm


SECTOR HEADER
Synch Signal 1
Enables reading speed of disk head to match rotational speed
Track Number
Identifies current track
Sector Number
Identifies current sector
Checksum 1
Provides error-check of header data
Gap 1
Separates sector header from data block
Synch Signal 2
File Link
Identifies next sector of current file
Data Block
128,256 or 512 bytes of user data, depending on the disk operating system
Checksum 2
Provides error-check of user data
Gap 2
Separates this sector from next
SECTOR DATA BLOCK

THE LOGICAL ANSWER

We have now covered some substantial ground in the Logic course, having concentrated in particular on the areas of basic logical principles and Boolean algebra. In this instalment, we take an overview of the rules explained so far, and provide a series of revision questions and answers.

Let us undertake a short review of the work we have covered in the previous instalments. The principles of logic apply to computers in the design of hardware that must carry out certain special jobs. We have already designed an adder circuit, which when combined with other adder circuits allows the addition of binary numbers (see page 48). This circuit models the human method of addition, allowing digits to be carried from one column to the next.

We used three basic elements in the design of this circuit, called logic gates. The functions that these gates could perform were indicated by the names we gave them (AND, OR and NOT), and were each defined by a truth table. The truth table is a simple way of writing down the output(s) from a circuit for any possible combination of inputs. Two inputs gave us four (2^2) possible input combinations, three inputs gave us eight (2^3) combinations, and so on. We also saw how logic gates can be linked together to give certain desired outputs. These more complicated circuits could also be described by their truth tables. In particular, we designed a five-gate Exclusive-OR circuit, which gave a true output when only *one* of its inputs was true (see page 32).

BOOLEAN ALGEBRA

The combination of logical elements can be described on paper by a set of symbols rather like that of normal algebra. The branch of mathematics concerned with the representation of logic is known as Boolean algebra, after the English mathematician George Boole (1815-64) who first defined its principles. Each of the three basic elements of logic has its own special symbol:

A AND B	A.B
A OR B	A+B
NOT A	$\overline{A}$

Just as there are laws that govern the manipulation of figures in arithmetic and letters in algebra, so there are special laws that govern the simplification of logical expressions. The laws of Boolean algebra are summarised in the table that follows.

SPECIAL RELATIONS	
Relation	**Dual**
$A.A = A$	$A + A = A$
$A.\overline{A} = 0$	$A + \overline{A} = 1$
$A.0 = 0$	$A + 1 = 1$
$A.1 = A$	$A + 0 = A$
$A.(A + B) = A$	$A + A.B = A$
$A.(\overline{A} + B) = A.B$	$A + \overline{A}.B = A + B$
DE MORGAN'S LAWS	
1) $\overline{A + B} = \overline{A}.\overline{B}$	
2) $\overline{A.B} = \overline{A} + \overline{B}$	
ASSOCIATIVE LAW	
$A.(B.C) = (A.B).C = A.B.C$	
$A + (B + C) = (A + B) + C = A + B + C$	
COMMUTATIVE LAW	
$A.B = B.A$	
$A + B = B + A$	
DISTRIBUTIVE LAW	
$A.(B + C) = A.B + A.C$	

Using these rules it is possible to simplify logical expressions and reduce the number of gates required in the final circuit. In addition to the algebraic method, we have also discussed the use of Karnaugh maps in logic circuit simplification (see page 92). Karnaugh maps represent a significant advance. Although they do not replace algebraic simplifications, they do reduce the amount of effort inherent in dealing with Boolean algebra. These maps, which are really extensions of Venn diagrams (see page 46), allow the grouping of expressions extracted from a truth table into twos, fours or eights. These groups represent simpler Boolean expressions, and so simplification is achieved. In practice, a combination of k-maps and algebra is often required to produce the most efficient circuit.

We have also discussed the use of the logical AND and OR operations in BASIC programming. We saw how these commands can be used to combine conditions in BASIC IF... THEN statements, and how they enable the programmer to turn on and off isolated bits within a register (see page 66).

As the culmination of this first stage of the course, we used all our knowledge of truth tables, Karnaugh maps, Boolean algebra and logic gate notation to design circuits that would give the desired outputs for certain defined tasks (see page 106). The following Review Exercise covers all the aspects of the course so far. Once you feel confident in using these rules, we can proceed to tackle more advanced logical theory.

Review Exercise

1) A pop group are guaranteed a hit single if they produce a good video to go with the song *and* their record company includes a free gift with the record, *or* if they appear on Top of the Pops. Draw a truth table to show all the possible outcomes. If each event is equally likely, what are the chances of the group having a hit single?

2) A school is setting up a chess club and membership cards are to be given out. The committee decides that membership should be restricted so that members must be:

i) A member of staff
ii) A fourth or fifth year pupil studying maths and science
iii) A sixth form pupil studying maths or science

Design an automatic membership card dispenser that is operated by pressing the buttons that best describe the prospective member. These buttons are:

A = Fourth year pupil
B = Fifth year pupil
C = Sixth year pupil
D = Member of staff
E = Pupil studying maths
F = Pupil studying science

Draw a circuit diagram for the machine.

3) A certain microcomputer has a register with address 23148 (decimal) that is used to control the video display. Bit 0 is the least significant bit in the register and bit 7 is the most significant. The screen can be switched into high resolution mode by setting bits 4 and 5 to one. As the other bits in the register are used to control other functions it is important that their values are not altered. Write BASIC commands that will:

a) Turn on the high resolution screen
b) Turn it off again

4) A bank's strongroom lock is based on key-operated switches. The manager, his deputy and the chief cashier all have keys. The door to the strongroom can be opened by two out of the three keys, subject to the following restrictions:

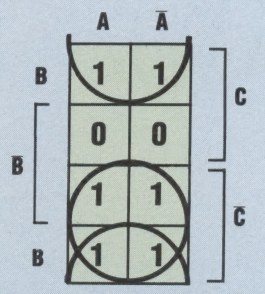
a) Between the hours of 9am and 5pm on weekdays the strongroom can be opened only in the presence of the chief cashier
b) At other times the strongroom can be opened only by the manager in conjunction with one of the other two people

Design a circuit to control the switching system.

5) In a binary digital data transmission system, where a high degree of reliability is required, each bit of data is fed simultaneously to three parallel channels which, at the receiving end, are fed into a 'vote taker'. If no transmission errors have occurred then all three bits should be the same. Under fault conditions the vote taker will correct an error in one channel by giving a single output that corresponds to the majority of the inputs. Design a circuit for the vote taker.

Answers To Exercise 5 On Page 107

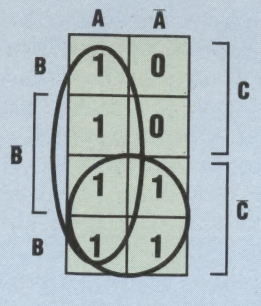
1a)



$= \bar{C} + B$
(alternatively $= \overline{C.\bar{B}}$)

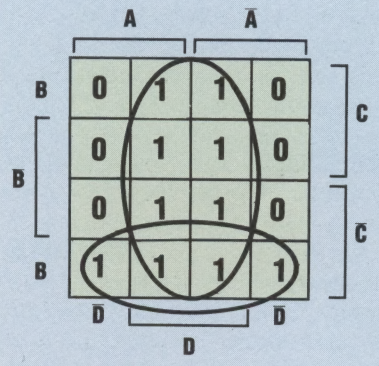
b)

$\overline{B + \bar{C}} + B.\bar{C} + A.C$
$= \bar{B}.\bar{C} + B.\bar{C} + A.C$ (de Morgan)



$= A + \bar{C}$
(alternatively $= \bar{A}.C$)

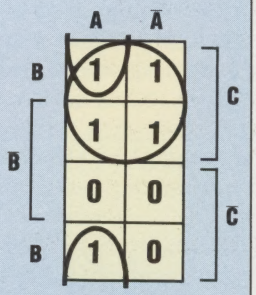
c)



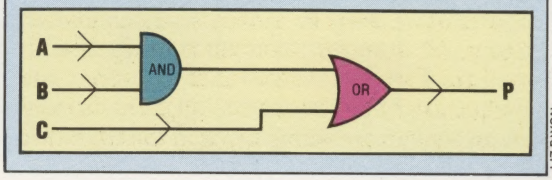
$= D + B.\bar{C}$

2) The truth table is:

Decimal	A	B	C	P
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



From the k-map we get the expression $P = C + A.B$. The resulting circuit is:



LIZ DIXON

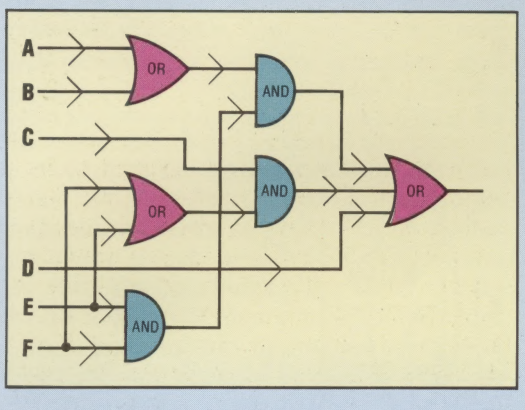
Answers To Review Exercise

1)

Good Video	Free Gift	Appear On TOTP	Have Hit Single
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Probability of a hit single = **5/8 = 62.5%**

2)



3 a) To turn high resolution on:

POKE23148,PEEK(23148)OR48

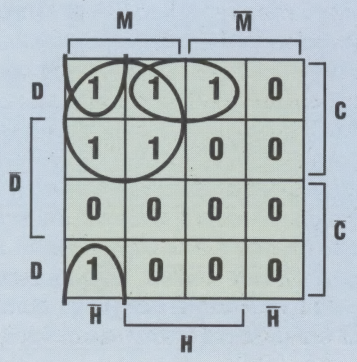
b) To turn high resolution off:

POKE23148,PEEK(23148)AND207

4) The truth table is:

Manager	Deputy	Chief Cashier	Hours 9–5	Unlock Door
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

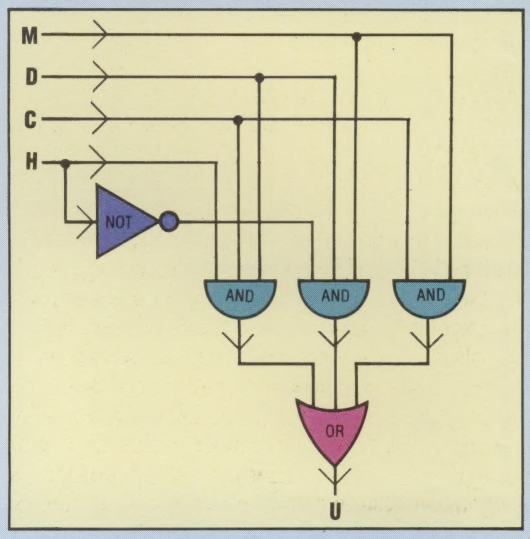
From the k-map:



We obtain the expression: U = M.C + M.D.$\overline{H}$ + D.C.H. It is easy to check if we are correct at this stage by converting this expression back into English. The expression says 'The door may be unlocked by:

i) the Manager and Cashier at any time
ii) the Manager and his Deputy out of hours
iii) the Deputy and Cashier between the hours of 9am and 5pm

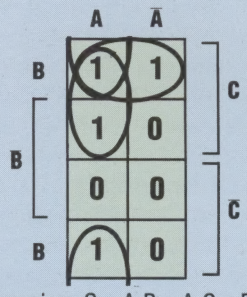
The corresponding circuit looks like this:



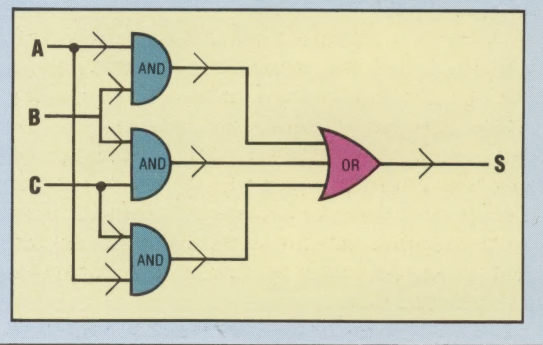
5) The truth table is:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

From the k-map:



We get the expression: S = A.B + A.C + B.C.
The corresponding circuit is:



NOXIOUS

B

BACKGROUND

If you are using a word processing package to create a document, and are typing continuously, then you may be surprised to learn that you are probably using the computer at less than 10 per cent of its efficiency. You probably can't manage more than ten keystrokes per second, yet the computer could process more than ten times that number — scanning the keyboard, interpreting the key pressed, storing it in RAM and displaying it on the screen. The result is that the computer spends approximately 90 per cent of its time simply executing a loop that waits for you to press a key.

There are obviously gains to be made if, while it was waiting, the computer could be put to good use elsewhere, and this is where the concept of *background* processing comes in. The computer executes one task in the foreground, which the user sees on his screen and interacts with, while another task is carried out in the background. The user doesn't interface with the background task, except at the start and finish of the job.

It follows, of course, that only certain types of job are suitable for processing in the background. One of these is sorting — a job that requires a lot of computing power, but where the user isn't directly involved. Another is printing: a background system would allow you to create or edit one document, while printing an already created one in the background. Another computer-intensive task that could be done in the background is the posting of invoices into a sales ledger.

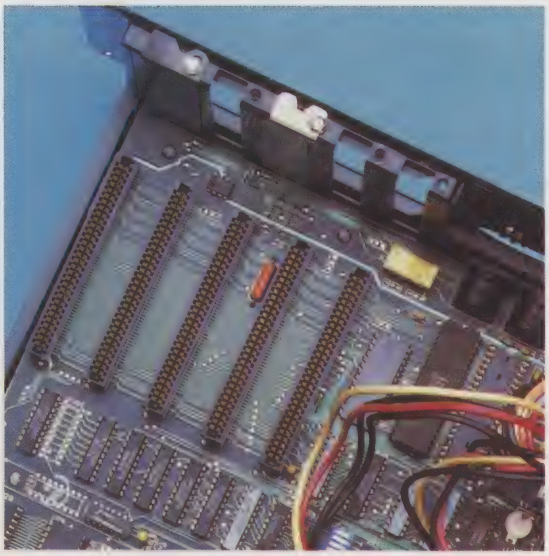
A computer-intensive job like sorting won't execute as fast in the background as it would if it had the whole computer to itself, but this isn't always a disadvantage. Most foreground-background systems are based on timesharing: the processor spends, say, 1/100th of a second seeing to the foreground, then 1/100th of a second in the background, and so on. Large mainframe computers can share their time between hundreds of programs and hundreds of users at once.

A few microcomputer applications programs have such a system built-in, but generally it is necessary to buy a special operating system (referred to as a 'multi-tasking' or 'concurrent' operating system). There is, for example, a concurrent version of the popular CP/M system.

BACKPLANE

Most home computers are now sold as self-contained units and come with strong warnings about the consequences of opening up the casing. But the first generation of microcomputers were based on a quite different concept. The computer came with a casing, power supply, and something called a *backplane*, which was a kind of board or frame featuring several sockets that connected together any devices or printed circuit boards that were plugged in.

The advantages of the backplane system were



CHRIS STEVENS

not just that it was easy to expand to include additional memory or interfaces, but that you could more or less *configure* (that is, design and put together) the computer as you wanted it.

Some of the backplane concept has been incorporated into contemporary educational and business machines, such as the Research Machines 380Z, the Apple IIe and the IBM PC, for which a large range of add-on boards are available. One business system, the Tycom Microframe, takes this one stage further and permits the user to choose the processor used, as well as the RAM and interfaces.

All the boards and devices that plug into the backplane must use the same set of connections for the bus. One of the common standards among early machines was called the S-100 (so-called because it had one hundred available lines), and there was a cut-down version called the S-50.

BACKUP

The importance of the concept of backup to microcomputing cannot be overstated. A backup is a duplicate or reserve copy of information, which is made in case something goes wrong with the main one (i.e. the working copy). It is necessary when dealing with magnetic media such as cassette tape or disk because, although the reliability of these has improved over the years, all magnetic media are subject to mishap. They might get lost, have coffee spilt over them, be left near a powerful magnetic field, or simply suffer from *dropouts*, when small pieces of the magnetic surface flake off.

The nature of computing is such that a tiny error can be responsible for rendering an entire file useless. The regularity with which you make backup copies (using the utility commands or programs provided by the operating system) should be proportional to the loss you would suffer if the data became corrupted. It is a lesson that most computer users learn the hard way — there is nothing to describe the way you feel when the computer refuses to retrieve the file you have spent hours creating.

DRAGON COMES OF AGE

The Dragon 64 is simply an enhanced version of the original Dragon 32. Thirty-two Kbytes of memory and a serial port have been added, and a new range of disk-based software is also available. Although the Dragon 64 has maintained compatibility with existing programs, it is clearly intended for more serious business work.

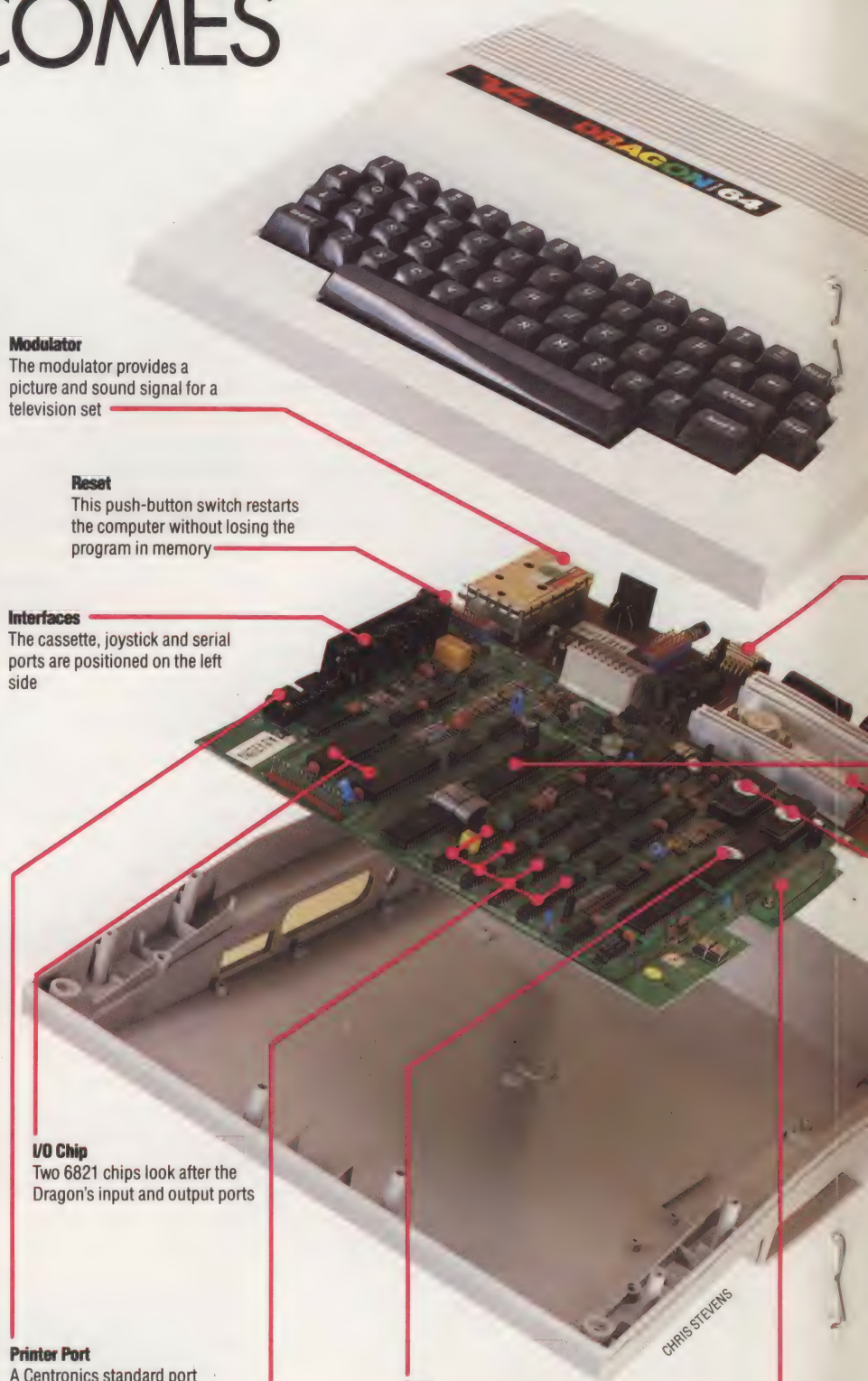
The Dragon 64 has the same keyboard as the later 32s, although it is of better quality than that of the earlier machines.But it lacks many useful keys, including Escape, Tab and Control.

When first switched on, the Dragon 64 — like the 32 — will automatically be in the '32 mode' in which it can run all Dragon software. BASIC users have up to 30 Kbytes of free memory. The command EXEC 48000 switches off the BASIC ROM and switches in the top 32 Kbytes of free memory. BASIC is then copied into high memory, leaving up to 45 Kbytes free. The full 64 Kbytes is available only under the other operating systems or machine code programs. Dragon DOS will reset the system to 32 mode, leaving only about 23 Kbytes free for BASIC programs.

Surprisingly, the 64's printed circuit board is very different from the 32's. It is the same computer, but the chips have been moved in order to make room for the new serial port and extra memory. As a result, 32 owners wishing to upgrade their machines will have to replace their machines with 64s, they will not be able to extend their 32s by plugging in extra components. The 64's CPU is the Motorola 6809 — an eight-bit design that arrived too late to gain the popularity of the 6502 and the Z80. A matching 6847 chip generates the display on either a television set or a composite video monitor. This chip gives the Dragon its rather curious display modes.

The Dragon works with a text screen of 32 × 16 characters on either a green or orange background. This limited display, while not a problem for the Dragon 32 user, has serious implications for those who want to use the Dragon 64 as a business machine. The new disk-based software, in particular the professional OS9 operating system, is severely restricted by the small display area.

For detailed graphics the Dragon has a number of high resolution modes, ranging from a resolution of 128 × 96 in four colours to a maximum of 256 × 192 in two colours. In comparison with many other machines this is very limited, particularly since the quality of display can be very poor. However, it does enable the


DRAGON 64
Modulator
The modulator provides a picture and sound signal for a television set
Reset
This push-button switch restarts the computer without losing the program in memory
Interfaces
The cassette, joystick and serial ports are positioned on the left side
I/O Chip
Two 6821 chips look after the Dragon's input and output ports
Printer Port
A Centronics standard port enables you to connect most types of printer
CPU
The Dragon is based around the Motorola 6809 microprocessor. This chip is very powerful and easy to program
Edge Connector
Used for both cartridge software and expansion, such as the Dragon disk interface
RAM
There's a full 64K of RAM on board, although only 45K is available to BASIC programmers
CHRIS STEVENS

Dragon Disk
Dragon's disk unit has either one or two drives of 175K each. The unit comes with the small but adequate Dragon DOS built into its interface cartridge

Power Input

Display Controller
The 6847 chip generates the Dragon's television and monitor displays

Heatsink
The Dragon's power transformer is in a separate box but the system still needs a big heatsink to keep cool

BASIC
These two EPROMs hold the Dragon's 16K Microsoft BASIC

Dragon 32
The Dragon 32 is still available and popular as a home computer. But many owners are upgrading their machines to the power and flexibility of the 64

IAN McKINNELL

Dragon to run arcade-style games, and can be used with the specially written software to provide an upper and lower case display of 51 columns by 24 rows — useful for spreadsheet and word processing programs.

STORAGE AND EXPANSION

The Dragon has a good dialect of Microsoft BASIC. This is an extended 16 Kbyte version and among its facilities are a set of powerful sound and graphics commands. The Dragon can retain a large number of graphics screen images in memory at once and can switch instantly between them, providing a simple way to do graphics animation in BASIC. You can also opt to use memory space normally reserved for graphics for BASIC programs and data.

The machine has a complete set of interfaces, including two analogue-to-digital converters and a standard Centronics port for connecting most types of parallel printers. The new serial port can connect to printers (including quality daisy wheel printers) or to other computers and equipment. An interesting way to develop the system would be to connect it to a professional terminal, allowing more advanced software to be used.

The Dragon interfaces to standard cassette recorders and can start and stop the tape from within the program and replay sound from the tape through the television speaker. BASIC also supports a range of cassette file handling commands. We have discussed the Dragon's disk drives (see page 104). These come with a small but workable operating system in ROM in the interface cartridge as well as a set of extensions to BASIC.

The Dragon 64 cannot be expanded into a serious business system until it has an adequate screen display and a professional keyboard. As it stands it is a very interesting and capable hobbyist's machine.

Multi-Tasking/Multi-User OS9
The Dragon 64's 6809 microprocessor, 64 Kbytes of memory and disk drives enable it to run the professional OS9 operating system. This is the leading operating system for 6809-based computers and provides some excellent facilities as well as access to a range of sophisticated business packages. Dragon has been to considerable lengths to make OS9 available for the 64. Unfortunately, the high cost of OS9 and its software, while quite in line with other business packages at around £80, may limit its popularity amongst home users.

OS9 provides the standard of facilities associated with UNIX, a minicomputer operating system that is available on the largest of business micros. Among its features are the ability for a user to run more than one program at once (multi-tasking), and for more than one person to use the computer at the same time (multi-user). To achieve these abilities, OS9 organises files of programs and information into a formal structure, rather than just as a straightforward directory of files on each disk. Each file also has passwords and access codes so that certain programs and information can be accessed only by particular people.

Most home users will not exploit these facilities and the Dragon would be under considerable strain if OS9 were used to the full. However, it does provide a low-cost way of experimenting with a sophisticated system, as well as giving the Dragon user access to advanced programs and alternative languages such as C and PASCAL

DRAGON 64

PRICE
Approx £225
DIMENSIONS
380×330×90mm

CPU
6809

MEMORY
64K RAM, of which up to 45K is available for BASIC programs
16K ROM

SCREEN
In the text mode, 16 rows of 32 columns, upper case only with a set of graphics shapes in eight colours. Graphics modes from 128×96 in four colours to 256×192 in two colours

INTERFACES
Joysticks (2), serial port, parallel printer port, cassette port, composite monitor with sound, TV and cartridge expansion port

DISK DRIVES
Up to two 175K disk drives, with either Dragon DOS or the OS9 operating system

LANGUAGES AVAILABLE
BASIC, FORTH, 6809 Assembly language. OS9 provides C, Pascal, structured BASIC and others

KEYBOARD
Typewriter-style with 53 keys

DOCUMENTATION
Unfortunately, the Dragon's manuals contain only the most rudimentary information and suffer from omissions and errors

STRENGTHS
The 64 has elements of a sophisticated computer — a large memory and a complete range of interfaces. The disk system is very good value and the ability to run OS9 software a major, if costly asset

WEAKNESSES
Suffers from limited keyboard and screen display modes, especially when used for serious work. The 1 MHz 6809 CPU may prove too slow for some OS9-based applications

SHOWCASE

The myriad pages of information on Prestel are created using a viewdata editing terminal. Such editors can cost up to £20,000 for a system consisting of minicomputer, digitising tablet and colour video camera. But the home computer owner can also get in on the act simply by spending ten pounds on a piece of software.

Apart from Prestel there are two other public viewdata systems in the UK: Ceefax and Oracle — although because they are broadcast through the air and not sent down a cable they are strictly called *teletext* systems. Many large companies run their own private 'in house' viewdata systems, which keep staff informed of company news and enable them to access specialised information — the latest stock market prices, for example. In a similar fashion, Rochford District Council have set up a free local viewdata system, containing 'What's On' information, details of council services, and so on.

In the High Street, viewdata is also becoming more common. Perhaps the travel agent booked your holiday using Prestel, or maybe the local video rental shop has a viewdata system in the shop window, working its way through a series of advertising pages — the so-called *carousel* system.

In all these areas it is possible to set up as an *Information Provider*, or *IP* for short. This means you create pages of viewdata information and then either use them on your own private viewdata system, or alternatively sell them to customers. For example, to put a single page onto the Prestel system costs between 15 and 45 pounds, but having done this you can then charge Prestel users every time they read your page — normally about one to 10 pence per access. Judging by the number of different IPs on Prestel there is obviously money

Page Creation

Let's go through the steps in creating a viewdata page. Assume that a local travel agent has commissioned you to create an eye-catching advertisement for a holiday in Greece, which with other pages could be put onto a carousel system in the shop window.

The first step, before we even touch the keyboard, is to draw a rough sketch of the page on paper. To be more precise, we could rule a 40 by 24 grid on the paper first and then translate each square on the paper into a screen character. We shall draw a typical holiday scene (beach, sea, sun etc.), and then superimpose the details of price, location, and so on

Page 100
Here we have drawn the sea and beach. The sea is produced by pressing the blue function key followed by the background key at the beginning of each line. Similarly, several lines of yellow background make up the beach

Page 101
We now start to 'build up' our hill out of green mosaic characters superimposed on the sea. In a similar way, the beginnings of an umbrella have been made with magenta mosaics

Page 102
The picture is now beginning to take shape: the hill has been finished, red stripes have been added to the umbrella and work has started on the sun. Each large letter is made up from six mosaic characters

Page 103
The picture is almost finished. The large letters, the sun and the umbrella have been completed, and a yacht has been added. For that authentic touch, a Greek temple now overlooks the sea!

Page 104
The completed page. To finish off, the text has been typed over the picture. 'TWO WEEKS IN GREECE' is done with double height characters, and if this were not a still photo we could see that the price is actually flashing. Incidentally, the 'waves' added to the sea are cyan minus (–) signs

IAN McKINNELL

to be made, either directly through this frame charge or indirectly through the publicity gained from it.

One enterprising group of students in South Wales approached a local department store at Christmas time and offered to set up a local viewdata system displaying pages of advertisements to the customers. They created the pages using a simple viewdata editor package on their micro and then installed the computer in the store's main window, where it automatically cycled through the various pages. By charging the store for the pages and for the micro rental, they earned themselves a useful Christmas bonus!

These are just some of the examples of ways in which viewdata can be used, but anyone with imagination could think of many others. All that is otherwise needed is a microcomputer, a little artistic talent and, of course, the software.

The specific viewdata package we shall look at here is called Viewtext and runs on the BBC Micro. For those particularly interested in using viewdata, this computer scores over most other home micros because of its built-in viewdata character set. Even though this package is one of the cheapest — it costs only ten pounds — it incorporates both a viewdata editor to create the pages, and a program to set up a 22 page carousel system.

The editor works much like any other on-screen editing system: the cursor is positioned using the cursor control keys and then the text is typed in. To select the various viewdata attributes — the colours for the text and background, flashing characters, double-height characters, etc — you just need to press the red function keys at the top of the BBC's keyboard. For example, to get the word 'HELLO' flashing and in yellow letters, you press the keys f3 (for yellow) and f8 (for flashing), and then type the word in.

At this point it is worth mentioning two details regarding the operation of a viewdata editor. Firstly, the attributes, such as 'flashing' or 'red text', are lost when you move down to the next line — the text reverts to small white letters on a black background. Secondly, each attribute takes up one invisible character space on the screen. Therefore, in the example above, there would be two apparently blank spaces in front of the word 'HELLO'. If you delete or type over these spaces then the relevant attribute will be lost.

VIEWDATA GRAPHICS

What has been said so far enables you to create pages consisting of text only, which is adequate for many viewdata applications. However, diagrams, advertising material and the like really require the use of graphics as well as text. Viewdata graphics are made up from a set of graphics characters, each character consisting of up to six pixels on a two-column by three-row grid. Pictures are built up in a similar way to that in which a mosaic is constructed from little tiles, and for this reason the characters are known as *alpha-mosaics*. The quality of the resulting picture is quite crude when compared with the graphics capabilities of most home computers, but with a bit of imagination any picture can be realised in alpha-mosaics.

The Viewtext Editor
The Viewtext package includes a crude editor to enable you to design your own teletext pages. In this example, the Greek holiday advertisement is being adapted by changing the text and the picture

Incidentally, the reason why the graphics are so basic is that each viewdata page is allowed only one Kbyte of memory, whereas a high resolution graphics screen on a home computer can use around 20 Kbytes. With the baud rate at which Prestel works, the delay in receiving a complete high resolution page would be unacceptable.

Using the alpha-mosaic characters is one area where the Viewtext editor does not perform well. The characters are obtained by first pressing the red function key (f9) to turn on the graphics, and then pressing any of the lower case or number keys on the keyboard. Each key produces a graphics character, but unfortunately there is no connection between what is engraved on the key and the character which appears on the screen — you have to continually refer to a table in the manual to find the required key. This process is very wearisome — the mosaic graphics in the accompanying photographs took a couple of hours to create. More sophisticated editors let you construct each character by turning on or off each of the six pixels in the character matrix, which is a much quicker method.

There are two special attributes that are only used with the mosaic graphics characters: Separate Graphics explodes each character so that the pixels are slightly separated from each other; Hold Graphics is used to cover over the blank spaces left when a new attribute is set. For example, to get a red mosaic character next to a green one without a blank space between, you would press the Hold Graphics function key before pressing the Green Graphics key.

That covers most of the features found on any viewdata editor; the more expensive versions have facilities to speed up the page creation process, which are useful in a commercial environment but not necessary for the home user. For example, a more expensive package would let you move or copy parts of the viewdata page, and would have a much quicker method for creating the mosaic graphics than those explained above.

Having created your page of viewdata, the next

High Resolution Viewdata

IAN McKINNELL

Most viewdata systems use alpha-mosaic graphics, which are economic in terms of memory but less successful when it comes to the quality of the pictures obtained. The alternative viewdata graphics standard is called alpha-geometrics and is capable of producing much more realistic pictures.

As well as the usual letters, numbers and mosaic characters, alpha-geometric graphics let you draw lines, circles and similar shapes using simple commands — in much the same way as the BASIC high-resolution graphics commands available on most home micros.

To accommodate the higher resolution, the screen is divided up into 320 by 240 pixels, as opposed to the 40 by 24 squares used in alpha-mosaic graphics. This does have the disadvantage that a very detailed picture — for example, a facsimile of a person's face — takes a couple of minutes to be transmitted down the phone lines.

One commercial system available in the UK is the MUPID terminal, costing £850. It is a dedicated microcomputer and is capable of displaying both alpha-mosaic and alpha-geometric graphics. Its novel features include the ability to produce animation on a single page, as well as half-tone colours and shading.

The alpha-geometric system is excellent for presenting detailed pictures such as maps or facsimiles. Unfortunately, it has not really caught on in the UK, although the Canadian viewdata system has adopted this standard

step is to save it on disk or tape, give it a page number and put it on your own viewdata system. The Viewtext package can hold up to 22 pages in the BBC's memory at any time, and each page is given a number between 100 and 121. When you have created and saved sufficient pages, you can run the carousel program. This loads in all the pages and then works through them in a repeating cycle, with you deciding on the time delay between each page. On the top line of any viewdata page appears such information as the date and time, the number of the page being viewed, and the name of the viewdata service — Prestel, Jones Travel, etc. The carousel program will ask the user for this information before cycling through the pages.

More sophisticated viewdata packages enable you to transmit and receive pages from other systems, using a modem and a telephone line. They also let you set up a proper database, similar to Prestel, with menu pages and access routes between the various pages, in addition to the simple carousel method of displaying pages. These packages tend to cost several hundred pounds, and require a lot of disk storage for the many pages of information, so they may be beyond the means of most home computer owners.

As the use of viewdata becomes more widespread, the demand for information pages will also increase. Running a simple editing program on a home micro, you can create viewdata pages to rival those produced by any commercial agency, and thus take advantage of this system.

MEMORY SPACE

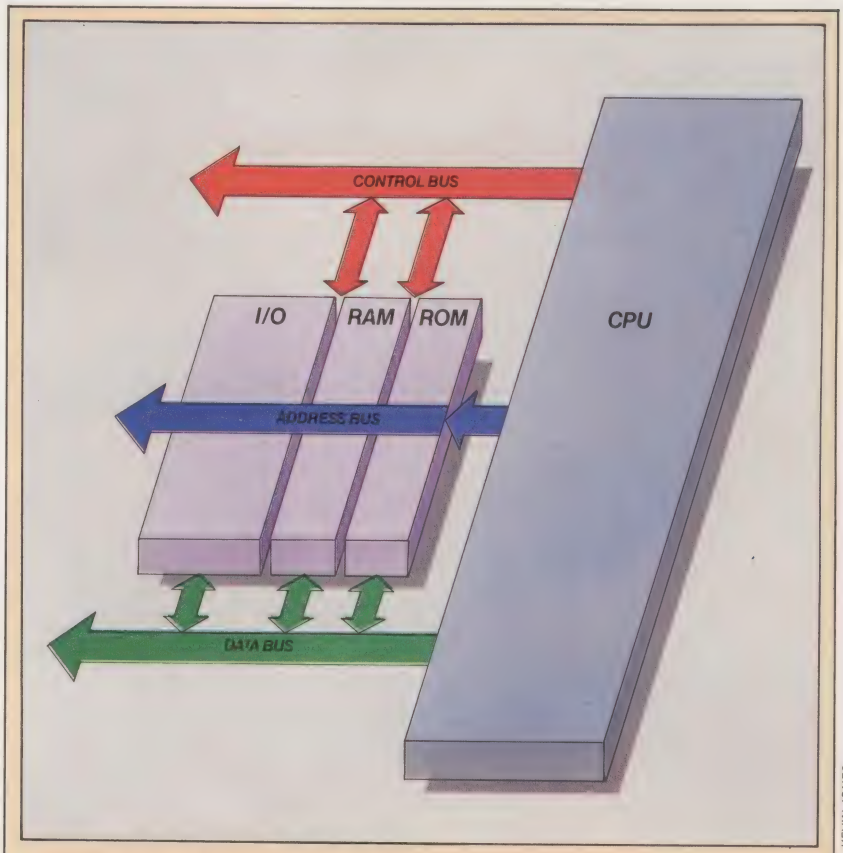
At this stage in the course, we look at ways of finding or reserving space in memory to store our machine code programs. We also take our first look at how we can perform a simple arithmetic task by using machine code instructions to manipulate the contents of the accumulator register in the Central Processing Unit.

In the last instalment of the Machine Code course, we developed a very simple Assembly language program, translated (assembled) it into machine code, loaded it into memory and executed it. We used the Monitor program on page 118 for the latter two tasks. If the program was a more sophisticated package containing an assembler, we could have used it for assembling our machine code program as well. At this stage in the course, it's no great hardship to do the assembly by hand — indeed, it's very educational. But once you've grasped the principles of the process, and as your Assembly language programs get longer, there won't be any point in concerning yourself with the actual machine code translation. In fact, with larger programs, assembling by hand gets very tedious, and is prone to error. Consequently, when you reach this stage in learning machine code, you may want to acquire an assembler program suitable for your machine.

There were many significant points about using the short machine code program that we gave (see page 117). We used one of the CPU registers to manipulate memory, we had to decide where in memory to store the machine code, and we caused the microprocessor to execute it. These are all aspects of Assembly language programming that particularly puzzle a beginner, and it's worth looking at them more closely. Let's start with the question of where to store the machine code.

To the CPU the only difference between one byte of memory and the next is whether they're read-write memory (RAM), or read-only memory (ROM). ROM chips contain system programs and data that must be protected from accidental or deliberate overwriting, and therefore can only be read. ROM can't be written to, so we can't load a machine code program into ROM. Those areas of memory apart, there's theoretically nothing to stop us loading a program into any other part of memory, but there are practical considerations that prevent us using some areas.

The CPU uses certain sections of RAM for temporary storage in the course of its operations, and if we load a program there, then either it will simply be corrupted by the CPU's overwriting it, or (and this is more likely) the CPU will read our machine code as if it were some of its own data. The operating system also uses large parts of RAM for storing its working data, and for running the computer system. Loading machine code programs (or anything else for that matter) into any of these areas would be unwise or impossible for the same reasons that prohibit use of the CPU's workspace. Furthermore, BASIC programs can take up all the remaining RAM — partly as program text and partly as variable storage areas. Once again, it's unwise to tamper with these areas, and so the programmer



KEVIN JONES

Small System Architecture

A typical computer system, in its most schematic form, comprises memory and a CPU. The former is made up of ROM chips (containing system programs), RAM chips, and specialist chips dedicated to input/output operations.

Data and control signals flow into and out of the CPU and around the system along *buses*. These are routes — very similar to ribbon cables — which can carry a byte or more of data at a time. These buses may be uni-directional like the *address bus*, which only transmits in one direction, or bi-directional like the *data bus*, which can transmit in either direction. The *control bus* carries switching information around the system, opening and closing logic gates to direct the flow of data. The *address bus* carries a 16-bit address from the CPU to select one byte of memory, allowing data to flow along the eight-bit *data bus* into or out of the byte

is faced with an apparently insoluble dilemma.

One answer is not to use BASIC at all, freeing areas like the BASIC Program Text Area. But it's usually convenient to write at least part of a program in BASIC, using machine code subroutines only where BASIC is too slow — animating screen displays, for instance. If BASIC and machine code programs are to coexist in RAM, then we must steal some space and designate it for machine code. We can move the boundaries of the BASIC Program Text Area, or we can find temporarily unused sections of it or of the operating system RAM.

Moving BASIC's boundaries is very easy on the BBC Micro, since these addresses are held in the system variables PAGE, TOP, LOMEM and HIMEM. PAGE, for example, points to the start of the BASIC Program Text Area — which is usually at address $1200. If we execute the instruction:

```
PAGE=PAGE+500
```

then the operating system will store BASIC programs 500 bytes higher in memory, leaving a 500-byte area free for our machine code programs. The same effect on the other machines is achieved by POKEing higher addresses into the system pointers (see the memory maps on page 58). Alternatively, we could reserve space by lowering the address of the top of the BASIC Program Text Area. On the Spectrum, the command CLEAR followed by an address does exactly that. The only constraint on the amount of space stolen from BASIC in this way is that there will be enough space left for our BASIC program.

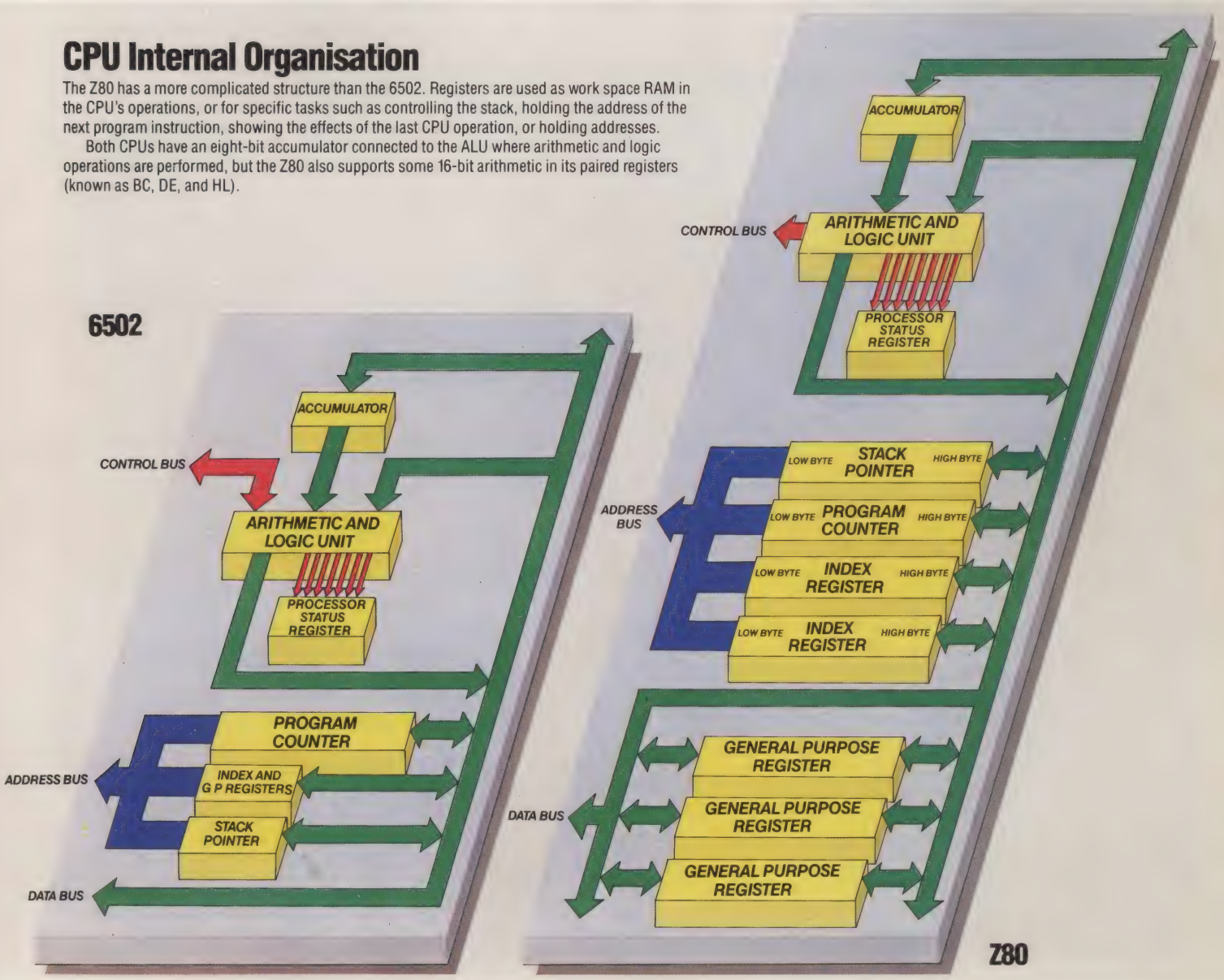
Small blocks of free space can be found in the operating system RAM, a typical example of which is the cassette buffer of the Commodore 64. This consists of 192 bytes of RAM (from $033C to $03FB) and is used by the operating system only when the cassette drive is employed. Many programmers find that this space alone meets all their machine code needs.

Even smaller blocks of space can be found within BASIC programs — REM lines, for example.

CPU Internal Organisation

The Z80 has a more complicated structure than the 6502. Registers are used as work space RAM in the CPU's operations, or for specific tasks such as controlling the stack, holding the address of the next program instruction, showing the effects of the last CPU operation, or holding addresses.

Both CPUs have an eight-bit accumulator connected to the ALU where arithmetic and logic operations are performed, but the Z80 also supports some 16-bit arithmetic in its paired registers (known as BC, DE, and HL).



KEVIN JONES

In the BASIC Program Text Area this line:

```
10 REM**************************
```

has 25 consecutive bytes containing $2A, the ASCII code for '*'. These bytes are never inspected by the operating system or the BASIC interpreter because to them the command REM means 'ignore the rest of this line'. Once the line has been entered into a program in this form, a machine code subroutine can be loaded into the asterisk bytes, where it will reside untroubled by the interpreter. The big advantage of this apparently messy method (often used in programs for the expanded ZX81) is that when you SAVE, and subsequently LOAD, the BASIC program, then the machine code subroutine goes along with it. Using the other methods described usually means having to save the machine code separately from the BASIC program. The trouble with this method is that LISTing the line causes the operating system to interpret the machine code bytes as ASCII character data, which may corrupt the screen display. This explains why there were warnings embedded in the Demonstration programs on page 19 for the BBC Micro and the Spectrum. The Commodore 64 version of that program loads the machine code subroutine into the cassette buffer, whereas the BBC and Spectrum versions load it into their opening REM line — hence the warnings about not LISTing these versions of the program.

Once you've written an Assembly language program, assembled it into machine code, and LOADed it into the RAM of your choice, then you can proceed to execute it. This is done through the BASIC commands CALL (BBC only), SYS (Commodore 64 only) or USR (all three machines). Each of these commands is followed by the address of the first byte of the machine code program, wherever it's stored. All three commands mean the same thing to the interpreter: 'execute the machine code program starting at the address given, and return to execute the next BASIC instruction when the RET or RTS op-code is executed'. It is similar to the GOSUB command in BASIC.

In the last instalment we wrote a program to copy one byte's contents into another byte by loading the accumulator with the contents from one address, and then storing the accumulator's contents in the other address. This illustrates the centrality of the CPU's role in the entire system: data and control must flow from memory, through the CPU, and back to memory. Whereas in BASIC we can write LET X=Y (meaning 'copy the contents of Y directly into X'), in Assembly language we have to copy into the CPU from memory, then out of the CPU back into memory. The CPU registers (see the accompanying panel) are the bytes of RAM inside the CPU itself where data from memory is stored or manipulated. Both the Z80 and the 6502 have a register called the *accumulator*, which is referred to by a majority of the Assembly language instructions, and is the register in which arithmetic is chiefly done.

Accumulator
This is really the central register of the CPU. Arithmetic and logic operations, as well as data transfers, are chiefly conducted via this register — more so in the 6502 than the Z80

Arithmetic And Logic Unit
Comprises a binary adder and logic gates, which permit access to individual bits of the registers and data bus. Properly controlled, it enables addition, subtraction and Boolean operations

Processor Status Register
Whenever a CPU operation is performed, the individual bits of the PSR show some of the effects of the operation — does it cause a zero result, for example, or is there a carry bit from an addition operation?

Program Counter
This points to the address in memory where the next op-code to be executed by the CPU is stored. The BASIC function USR(address) causes the address specified to be loaded directly into the program counter, so that CPU execution proceeds from that point

Stack Pointer
Carries the address of the next byte in RAM of free workspace for the CPU's use. Whenever the CPU writes some data to the stack, the stack pointer is changed to point to the next free byte

Index And General Purpose Registers
Index registers are used by the CPU to address memory in a variety of ways, while the General Purpose Registers are used as general workspace, and for specific CPU purposes

KEVIN JONES

Suppose we want to add the two numbers $42 and $07 (remember that the symbol $ means the number is hexadecimal). We simply put one of them into the accumulator, and add the other in on top of the first — their sum will literally 'accumulate' in the register. Here are the instructions to perform this:

Z80		6502	
LD	A,$42	LDA	#$42
ADC	A,$07	ADC	#$07

Here the Z80 instructions both refer to the numbers to be loaded and added, whereas in the 6502 version the numbers are preceded by #, which shows that they are actual numbers rather than addresses. Thus, LDA #$65 means 'load the number $65 into the accumulator', whereas LDA $65 means 'load the contents of the byte whose address is $65 into the accumulator'. Similarly, the add instruction, ADC (it happens to be the same mnemonic in both Z80 and 6502) means in this case: 'add an actual number into the accumulator'. The numbers $42 and $07 are said to be 'immediate data', and LDA #$42 may be read as 'load the accumulator with the immediate data #$42'.

After these two instructions have been executed, the sum of the numbers will be contained in the accumulator. It is 'invisible' to us there, so we must store the accumulator contents in a byte of RAM where it can be inspected. The program must end with a return instruction, and must begin with an instruction to put an associated CPU register into the correct state, so the full programs read as:

Z80	
Machine Code	**Assembly Language**
A7	AND A
3E 42	LD A,$42
CE 07	ADC A,$07
32 ?? ??	LD BYTE1,A
C9	RET
6502	
18	CLC
A9 42	LDA #$42
69 07	ADC #$07
8D ?? ??	STA BYTE1
60	RTS

We won't bother about the meaning of the first instruction in either program at the moment, but notice that the fourth instruction contains the symbol BYTE1, rather than an actual address. The value of BYTE1 will be different from machine to machine, so we'll just use the symbol here, and replace it by a real hex number when we come to assemble the code.

Now we must decide where to locate the machine code, and what address BYTE1 represents. Choose a place to store the machine code and then make BYTE1 equal to the address of the byte after the end of the program, and put that address in lo-hi form into the machine code. After that use the Monitor program on page 118 to load and execute the machine code, and to inspect the byte where the result — $49 — should be stored.

COMPONENT PARTS

Electronic equipment, of every kind, is made up of electrical components. Some of these components are quite dazzling in their complexity. The microprocessors and video control chips at the heart of every microcomputer are so complicated that the tremendous range of their actions and their minute size seem alien to the human world.

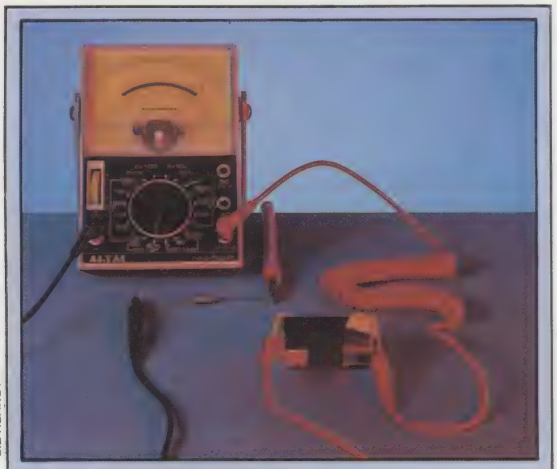
Even the most complicated of integrated circuits need support from, and are even largely made up of, much simpler electrical components. These simple building block components can be classified into two types: active and passive.

The passive components are the simplest. These merely hamper the electrical signal that flows through them. Different signals are hampered in different ways. That is what makes these components useful. Examples of such components are resistors and capacitors.

Active components are altogether more clever. They can actually add to the signal that flows through them. A transistor is a good example; when fed with a small signal it gives out a larger (amplified) signal.

Passive components are made up of simple materials. Apart from the packaging materials (the various resins and plastics) that you actually see when you buy, say, a resistor, the component is made of copper and steel and carbon. These are all conductors. The active components, though, are all made up largely of silicon or germanium. These two elements have special properties that make them behave not as conductors, like steel and copper, nor as the insulating resins and plastics, but in some way in between. Under some conditions they conduct electricity and under other conditions they do not. That is why these materials are known as semiconductors.

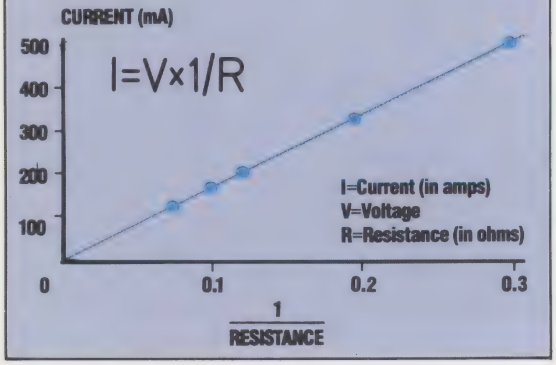
The active components, because they are based on semiconductors, work in many interesting ways. You cannot say that applying a particular voltage across a semiconductor will result in an electric current of a certain value flowing through it, nor indeed any current at all. All conductors, though, follow one very simple law. Ohm's Law (see page 114) describes the way in which any simple conductor will react to the electrical signal that flows through it. The transistor is the most important of these components to computing, as it is the basis by which computers can store and process information. In the next instalment we'll use transistors to build some of the simple logic gates that we've discussed in the Computer Science course.



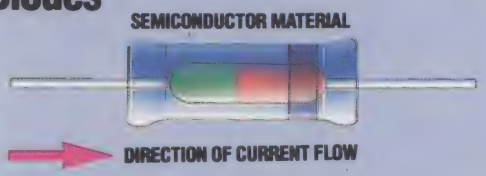
LIZ HEANEY

Proving Ohm's Law

You can prove Ohm's Law for yourself with only a few resistors, a battery, some wire and a multimeter. The resistors should be rated at one watt or greater, with values from 3.3 ohms to 15 ohms. Set the multimeter to read current. A 400 mA (milliamp) scale on the meter will be suitable. Now connect it into the circuit with the other components, as shown above. Insert each of your resistors in turn and check the reading on the multimeter. If you can now plot your readings of current against one divided by the resistor values, you will find that you get a straight line graph in which I is directly proportional to 1/R with a constant voltage. Therefore I=V × 1/R, which is V=I R — the formula for Ohm's Law



Diodes



Diodes are the electrical equivalent of a one-way valve. A diode has a very low resistance to electric current in one direction (usually a few ohms) and a very high resistance (several million ohms) to current in the other. This disobedience to Ohm's Law is because the diode is not an ordinary conductor but is a semiconductor made out of materials such as silicon and germanium. You can actually see the semiconductor in some glass packaged diodes. Every diode has a stripe at the end to which current can flow

Spectrum

```
 20 DIM b$(3,7)
 25 DIM c$(4,7)
 30 INPUT "input colour of bands"
 40 INPUT "band number 1",b$(1)
 50 IF b$(1)="gold" OR b$(1)="silver" T
    HEN  GO TO 30
 60 INPUT "band number 2",b$(2)
 70 INPUT "band number 3",b$(3)
 80 PRINT
 90 FOR i=1 TO 3
100  RESTORE 180
110 FOR j=0 TO 9
120 READ c$(1),c$(2),c$(3),c$(4)
130 IF c$(1)=b$(i) THEN  GO SUB 1000
140 NEXT j
150 NEXT i
160 PRINT "  ohms"
170 STOP
180 DATA "black","0","0",".0","brown","
    1","1","0","red","2","2",".00","
    orange","3","3",".000"
190 DATA "yellow","4","4",".0000","green
    ","5","5",".00000","blue","6","6","
    .000000"
200 DATA "violet","7","7",".0000000","gr
    ey","8","8",".0","white","9","9",".0"
1000 FOR k=1 TO 7
1020 IF c$((i+1),k)<>" " THEN  PRINT c$(
(i+1),k);
1025 NEXT k
1030 RETURN
```

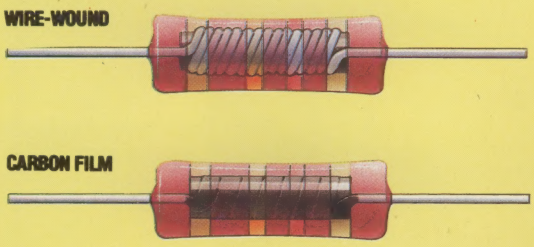
BBC

```
 10 REM resistor colour codes
 20 DIM B$(3),C$(3)
 30 PRINT "Input colour of bands:"
 40 INPUT "BAND NUMBER 1",B$(1)
 50 IF B$(1)="GOLD" OR B$(1)="
    SILVER" THEN GOTO 30
 60 INPUT "BAND NUMBER 2",B$(2)
 70 INPUT "BAND NUMBER 3",B$(3)
 80 PRINT
 90 FOR I=1 TO 3
100   RESTORE
110   FOR J=0 TO 9
120     READ C$(0),C$(1),C$(2),C$(3)
130     IF C$(0)=B$(I) THEN PRINT C$(I);
140     NEXT J
150   NEXT I
160 PRINT "  ohms"
170 END
180 DATA BLACK,0,0,, BROWN,1,1,0,
    RED,2,2,00, ORANGE,3,3,000
190 DATA YELLOW,4,4,0000, GREEN,5,5,
    00000, BLUE,6,6,000000
200 DATA VIOLET,7,7,0000000, GREY,8,8,,
    WHITE,9,9,,
```

For the Commodore 64 replace lines 40, 60 and 70 with:

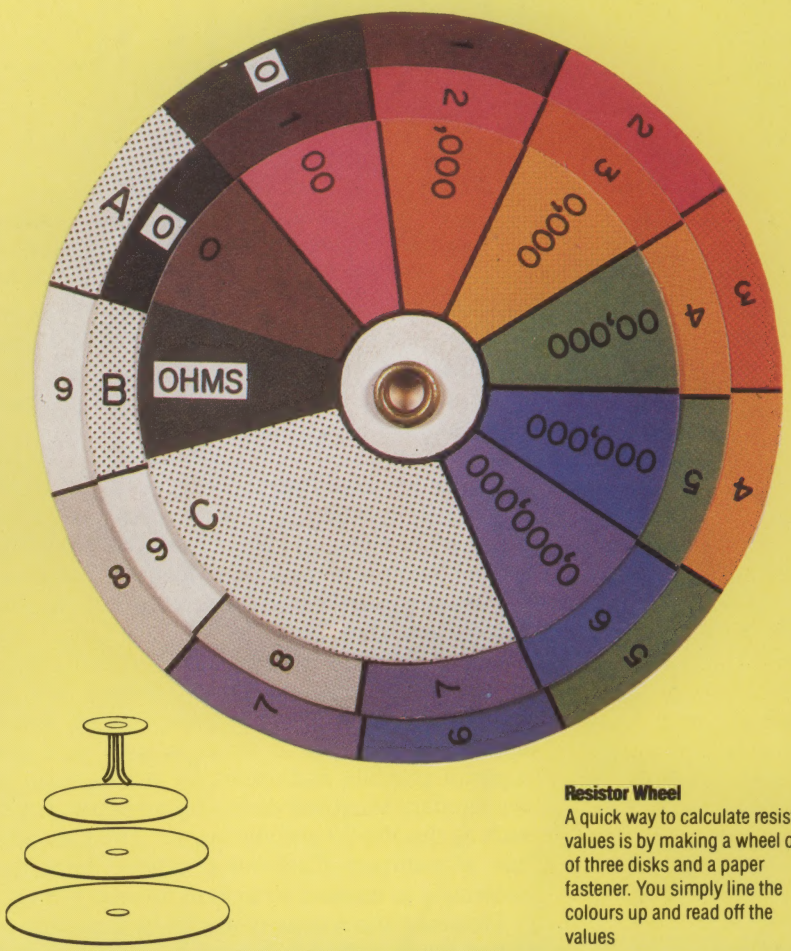
```
40 INPUT "BAND NUMBER 1
   ";B$(1)
60 INPUT "BAND NUMBER 2
   ";B$(1)
70 INPUT "BAND NUMBER 3
   ";B$(3)
```

Resistors



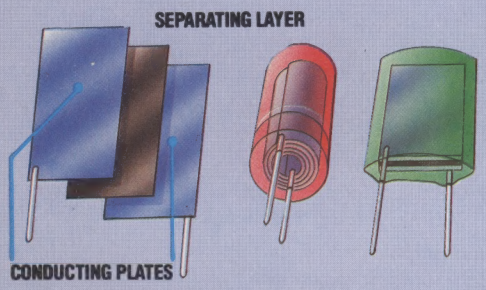
Resistors are one of the simplest electronic components — the two common types are wire-wound and carbon film. Wire-wound resistors are made from long lengths of wire wound tightly around an insulating cylinder and encased in an insulating cover. This current must flow along the entire length of the wire, resulting in a suitable reduction in the current. Carbon film resistors work in the same way, but the path for the current is a spiral cut in a carbon film around the insulating cylinder.

Both types of resistors are marked with coloured bands to identify their values. Reading these is a simple trick to learn. The first two bands represent a figure for the resistance in ohms and the third a value to multiply this figure by. The final gold or silver band tells you the tolerance the component is made to and can be used to tell which way round to read the other bands. You always read towards the gold or silver band. To save you learning the colours off by heart, a short program such as the one listed here can be used to calculate resistor values

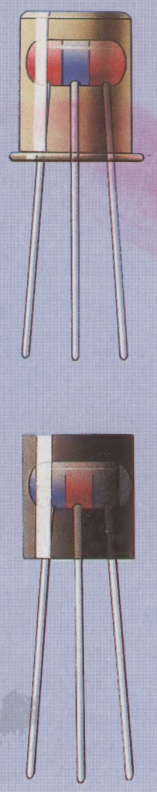


Resistor Wheel
A quick way to calculate resistor values is by making a wheel out of three disks and a paper fastener. You simply line the colours up and read off the values

Capacitors



Capacitors resist alternating current. High frequency AC currents flow through capacitors more easily than lower frequency ones, making them useful for cleaning up or filtering an electrical signal. Many microprocessor boards are dotted with capacitors serving just this purpose. Capacitors are essentially two conducting plates separated by an insulating layer either of various special salts (for high voltages) or of a ceramic material (for low voltages). The plates can be quite large and the actual size of the component is reduced by rolling the layers up into a tight spiral



Transistors

The transistor is the most complex of these components — it's a semiconductor device and its invention marked the start of modern electronics. A transistor has two basic uses. As an amplifier, it can take a small input current and produce a high output current. As a switch, one current can be used to turn another current on or off. This ability to act as an electronic switch is the basis of all digital electronics and is essential to the operation of computers

Like diodes, transistors are made of semiconductor materials but have two junctions inside them rather than just one. There are three leads to the semiconductors, usually called the base, collector and emitter. It is how these leads are fitted into the circuit that determines whether the transistor acts as an amplifier or as a switch. Unlike resistors and diodes, there are no standard ways of identifying the leads on a transistor or their specifications. The usual method is to resort to the manufacturer's reference book for that particular part number

KEVIN JONES

PAGE THE ORIC

Oric Products International was set up as a rival to Sinclair Research. It was financed by British Car Auctions and was staffed by people from Tangerine Computer Systems and Tansoft. However, in its early days, the company was beset with design faults and problems of supply.

When it was founded in 1982, Oric combined marketing experience — in managing director Barry Muncaster and sales director Peter Harding — with the design experience of Dr Paul Johnson in hardware, and Paul Kaufman in software. The machine the company developed, the Oric-1, is a 6502 equivalent of the Sinclair Spectrum. It comes in 48 Kbyte and 16 Kbyte versions and in comparison with the Spectrum has a better keyboard, comes in a stronger case, and has superior graphics and sound. Its resident BASIC is the standard Microsoft dialect found in computers such as the Vic-20 and the Apple. The Oric also has a standard Centronics printer interface, allowing it to connect straight to full-size printers.

However, the company was hit by a number of major snags, suffering from delivery delays and design faults. The early machines had problems with tape loading and unstable screen displays. The BASIC ROMs contained a number of small but irritating bugs, making programming awkward. In addition, the Oric's memory-saving graphics techniques made it very difficult to program. These problems were exacerbated when the company was forced to reprice the system above the Sinclair Spectrum.

As a result of these difficulties, the machine did not sell well at first and it was difficult to obtain software for it. Tansoft worked hard to support the machine, providing languages such as Assembler and FORTH, and now the Oric has a respectable software base.

Old And New Orics
The futuristic-looking Oric 1 lasted 15 months before Oric decided to repackage it as the smart red-and-black Atmos with a full keyboard and a revised BASIC ROM. The Oric four-colour printer/plotter also had a matching redesign

Oric also promised a whole range of add-ons, including disk drives, a modem and a printer. Of these, only the four-colour printer/plotter emerged. Following this shaky start, the Oric has slowly established itself. Overseas sales have been very impressive. Of the 170,000 machines manufactured in 1983, over 50 per cent were for export. The Oric is particularly popular in France, where it was voted top micro in 1983, because its RGB monitor output, unlike that of other British micros, will work on French television sets. In Japan, also, a specially designed Oric has sold well.

Fifteen months after its launch date, Oric cured many of the problems inherent in the Oric 1 in a very novel way. It launched the Oric Atmos model, which features a full keyboard and improved BASIC ROMs, but is otherwise the same machine repackaged. With the announcement of the Hitachi three-inch disk drive to complement the system, the Oric Atmos begins to look like an extremely good-value disk-based micro — more in league with the Acorn Electron and the Commodore 64 than the Spectrum.

Once again, however, initial optimism has been tempered with problems. There have been delays in the delivery of disk drive shipments and the Atmos has trouble loading and running existing tapes. But a new development centre in Cambridge has been set up and a tie-up with the British semiconductor manufacturer Inmos should provide a good base from which to work in the future.

Mentathlete

Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

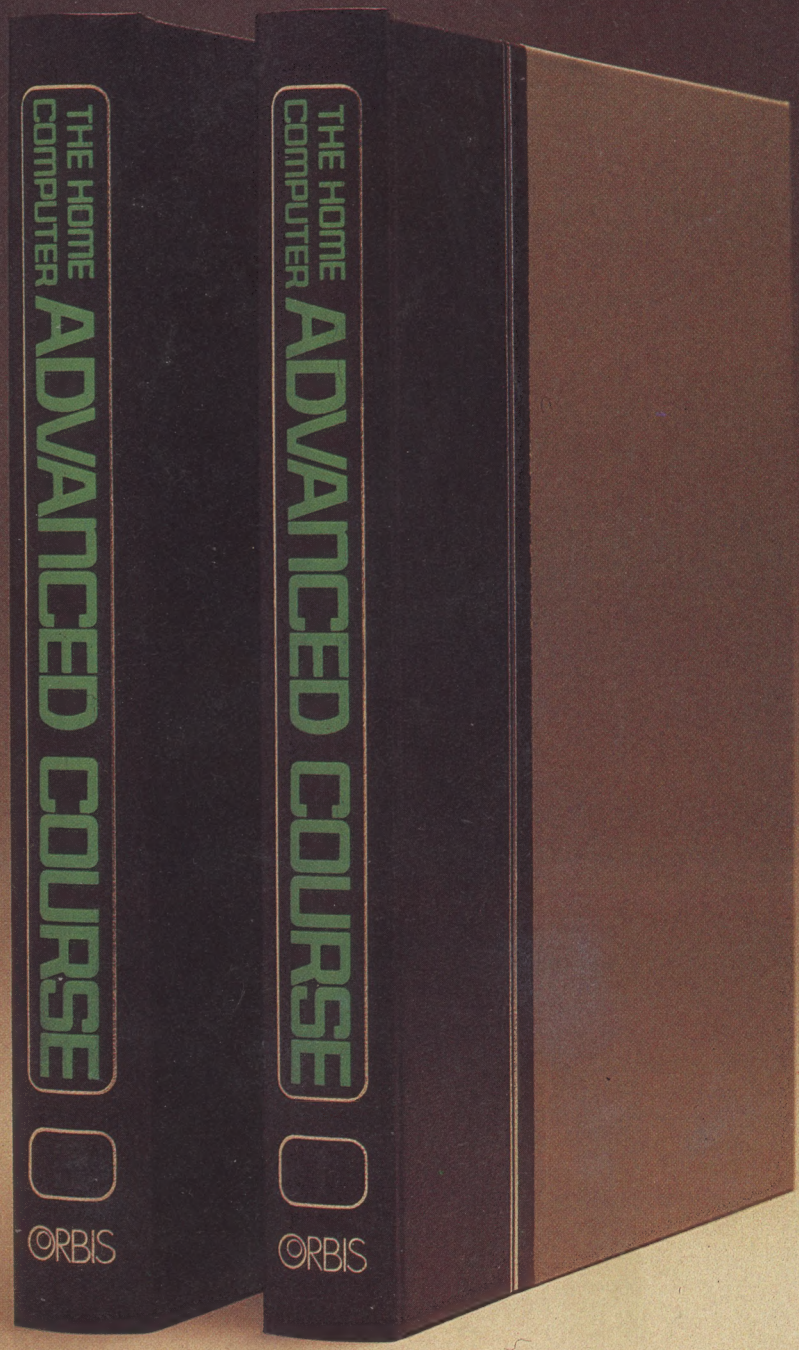
No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair

THE HOME COMPUTER ADVANCED COURSE

WE HAVE DESIGNED BINDERS SPECIALLY TO KEEP YOUR COPIES OF THE 'ADVANCED COURSE' IN GOOD ORDER.



All you have to do is complete the reply-paid order form opposite – tick the box and post the card today – **no stamp necessary!**

By choosing a standing order, you will be sent the first volume free along with the second binder for £3.95. The invoice for this amount will be with the binder. We will then send you your binders every twelve weeks – as you need them.

Important: This offer is open only whilst stocks last and only one free binder may be sent to each purchaser who places a Standing Order. Please allow 28 days for delivery.

Overseas readers: This free binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain binders now. For details please see inside the front cover. Binders may be subject to import duty and/or local tax.

The Orbis Guarantee: If you are not entirely satisfied you may return the binder(s) to us within 14 days and cancel your Standing Order. You are then under no obligation to pay and no further binders will be sent except upon request.

PLACE A STANDING ORDER TODAY.